Reinforcement Learning

Reinforcement Learning

Intelligence as <u>Search</u>

Intelligence as <u>Adversarial Search</u>

Intelligence as <u>Function Approximation</u>

Intelligence as…

Reinforcement Learning

- Russell & Norvig Ch 17: Making Complex Decisions
- Russell & Norvig Ch 22: Reinforcement Learning
- OpenAI - Spinning Up: Key Concepts https://spinningup.openai.com/en/latest/spinningup/rl_intro.html#key-concepts-and-terminology
- https://www.cs.ubc.ca/~murphyk/Bayes/pomdp.html A brief introduction to reinforcement learning - Kevin Murphy

Reinforcement Learning

<u>Search:</u>

Need good heuristics

<u>Adversarial Search:</u>

Need a good evaluation function

<u>Machine Learning:</u>

Need data examples to learn a good approximation

Reinforcement Learning

<u>Search:</u>

Developer writes heuristics

<u>Adversarial Search:</u>

Developer writes evaluation function

Imperative specification for *how* to solve the problem

<u>Machine Learning:</u>

Developer collects data

Implicit specification via examples

Reinforcement Learning

<u>Search:</u>

Developer writes heuristics

Imperative specification for *how* to solve the problem

<u>Adversarial Search:</u>

Developer writes evaluation function

<u>Machine Learning:</u>

Developer collects data

Implicit specification via examples

<u>Reinforcement Learning:</u>

Developer defines state space, actions, and **reward function**

Reinforcement Learning

<u>Reward Function:</u> Describes how “good” a state transition is

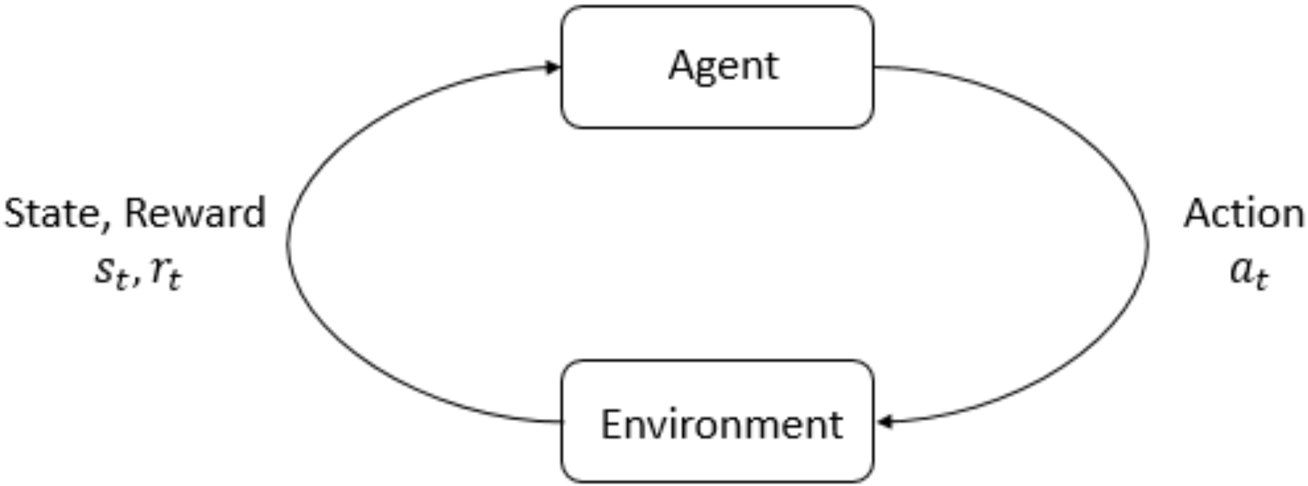
The reward function is an *abstract definition of good behavior*

Different from machine learning…
    ML relies on *concrete examples* of good behavior

Different from search in Weeks 1 & 2…
    Those approaches relied on specifications of *how* to do good behavior

Reinforcement Learning

RL involves an agent-environment interaction loop

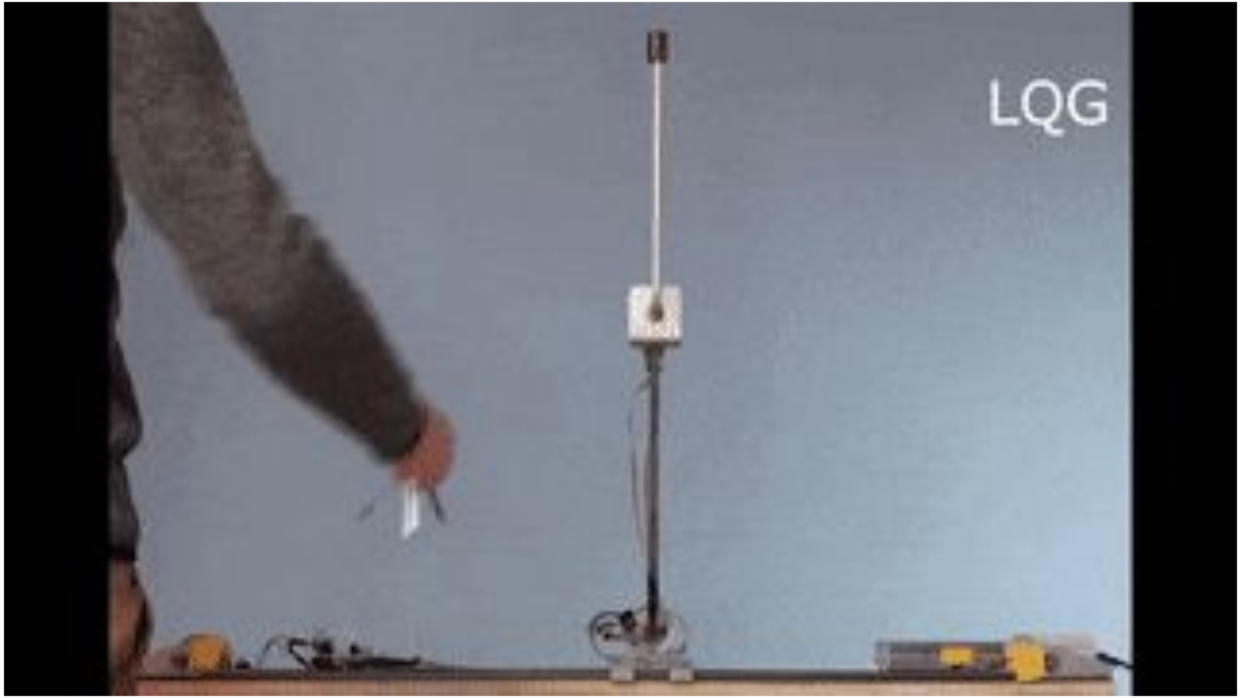


Markov Decision Process (MDP)

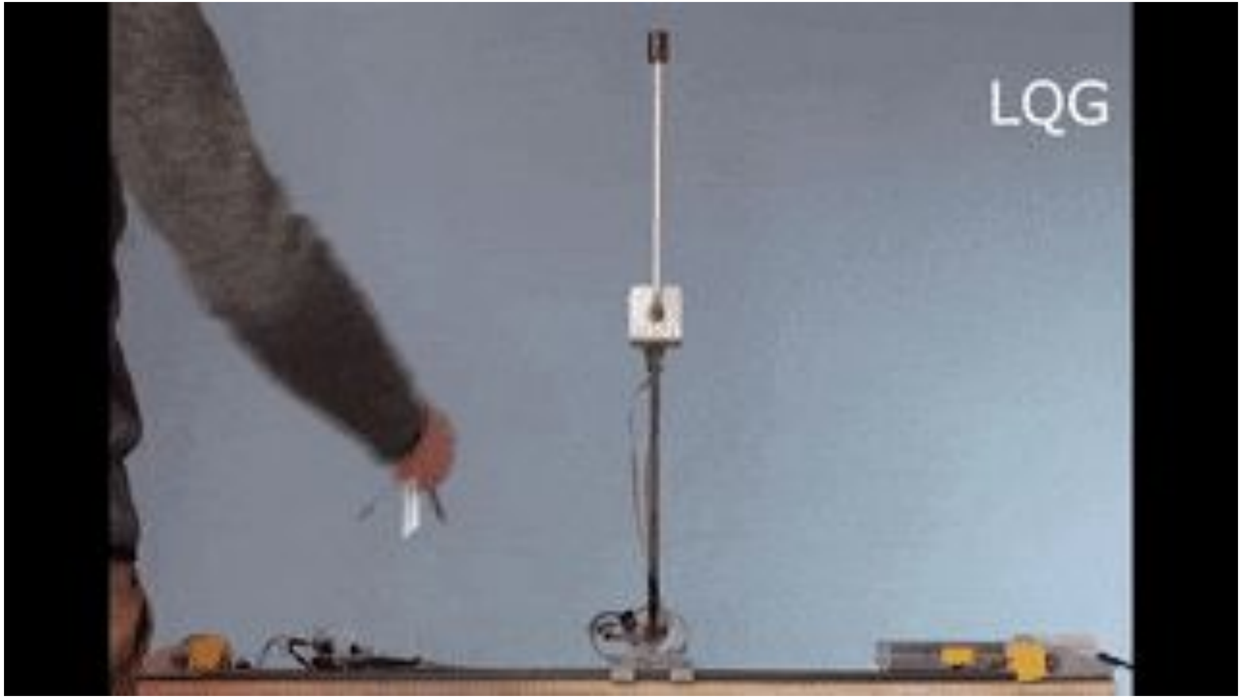
An MDP is a 5-tuple, $\langle S, A, R, P, \rho_0 \rangle$

S	set of valid states	
A	set of valid actions	
$R : S \times A \times S \to \mathbb{R}$	reward function	
$P : S \times A \to \mathcal{P}(S)$	transition probability function	$P(s' \mid s, a)$ is the probability of transitioning into state s' if you start in state s and take action a
$\rho_0 : S \to \mathbb{R}$	initial state distribution	$\rho_0(s)$ is the probability of starting in state s

Inverted Pendulum: A Case Study

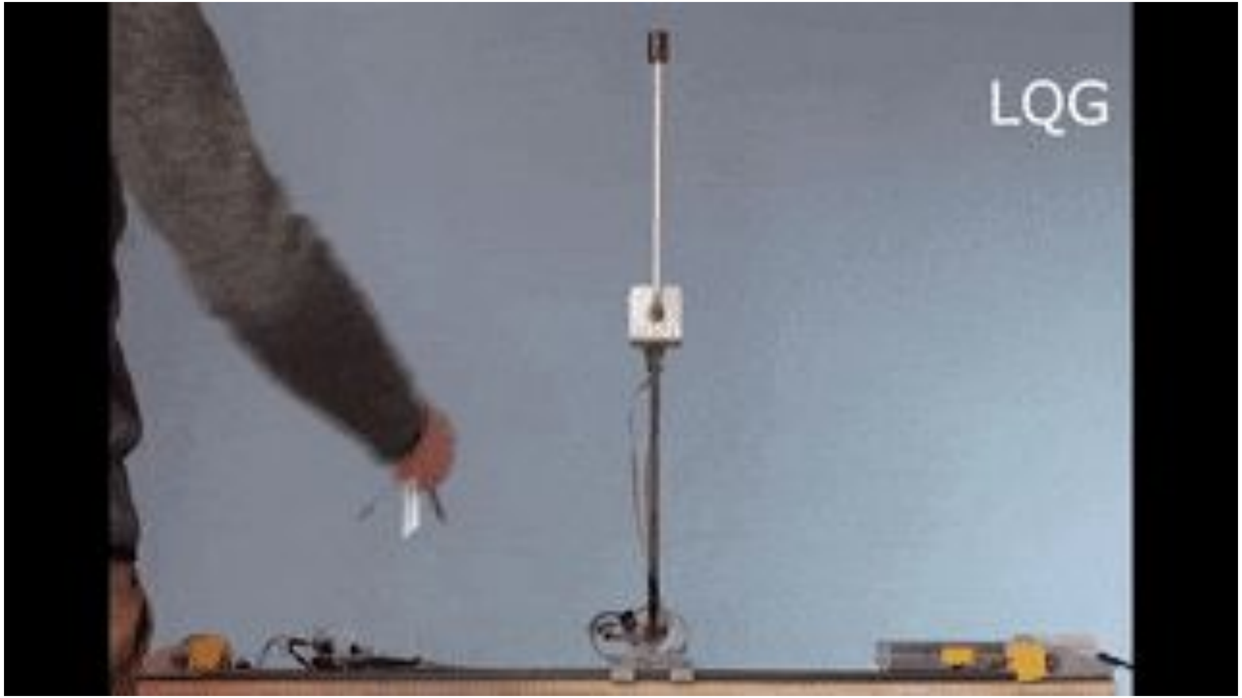


Inverted Pendulum: A Case Study



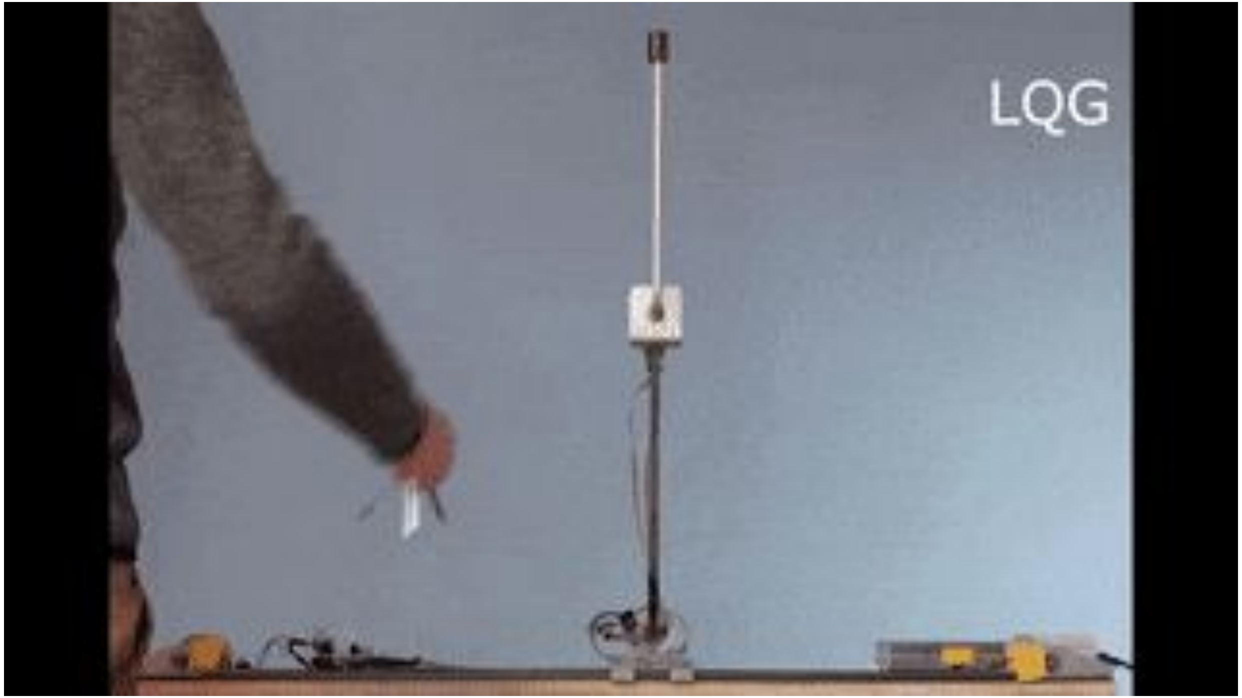
- States?

Inverted Pendulum: A Case Study



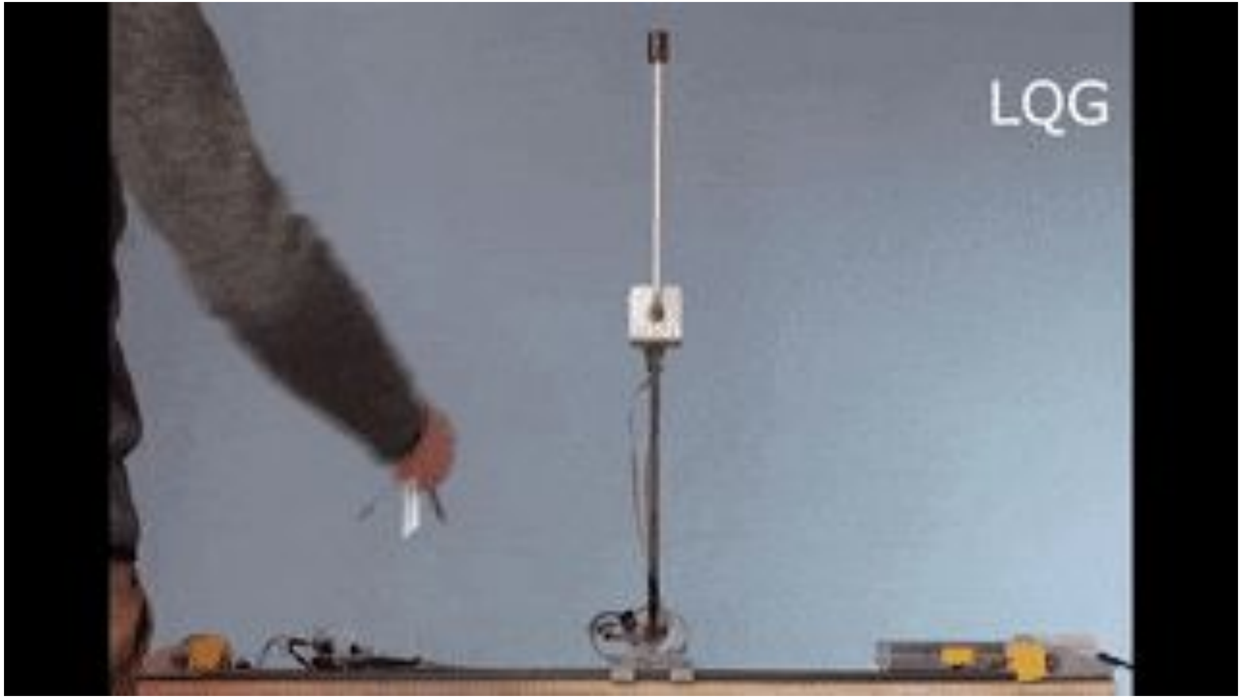
- States?
  - Roller position
  - Pendulum angle

Inverted Pendulum: A Case Study



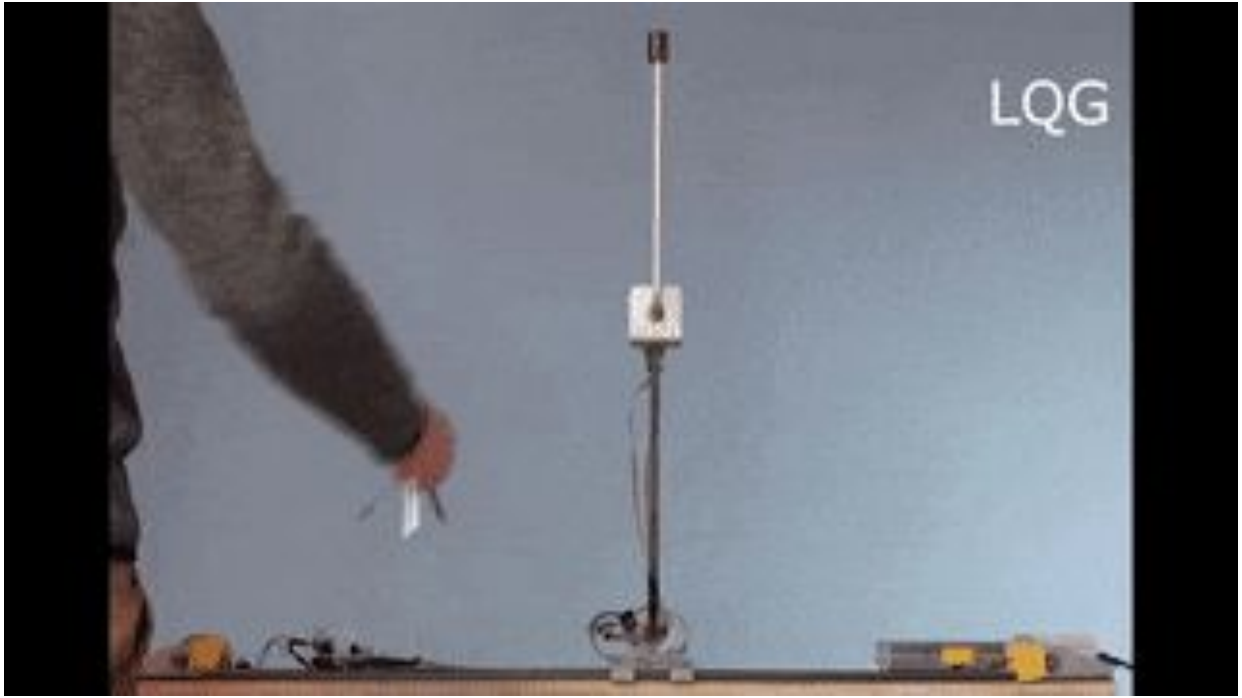
- States?
  - Roller position
  - Pendulum angle
- Actions?

Inverted Pendulum: A Case Study



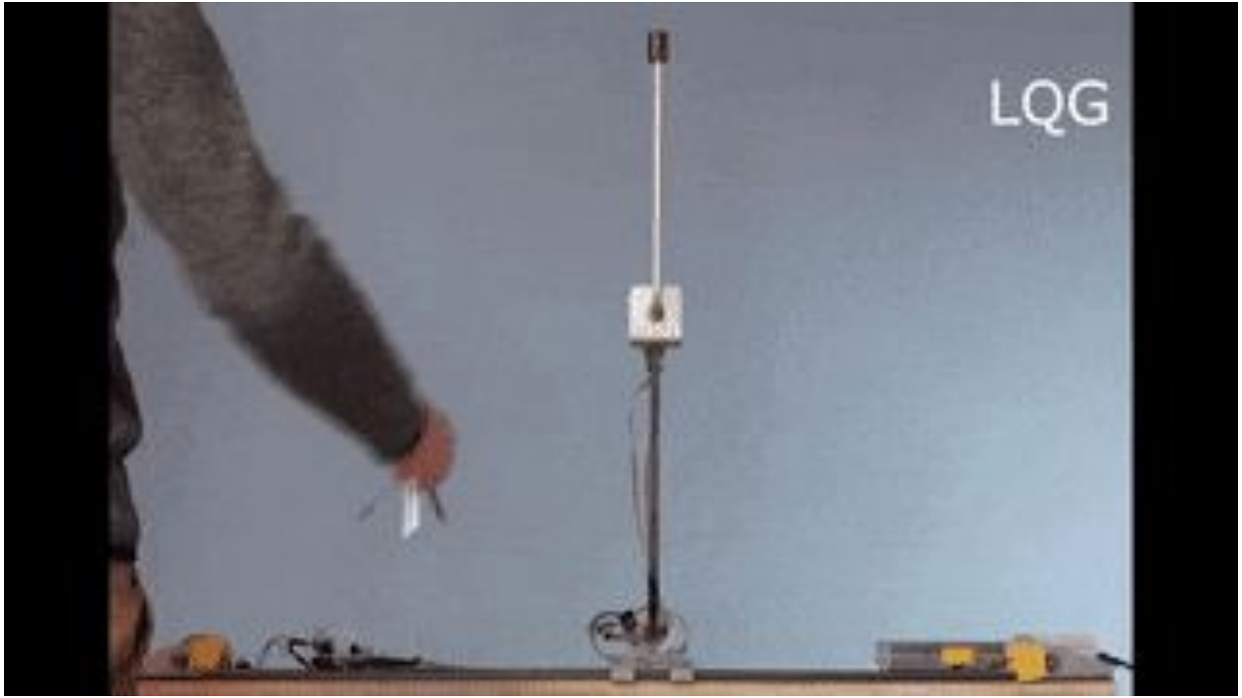
- States?
    - Roller position
    - Pendulum angle
- Actions?
    - Left/right

Inverted Pendulum: A Case Study



- States?
  - Roller position
  - Pendulum angle
- Actions?
  - Left/right
- Rewards?

Inverted Pendulum: A Case Study



- States?
  - Roller position
  - Pendulum angle
- Actions?
  - Left/right
- Rewards?
  - Angle

States, Actions, Rewards

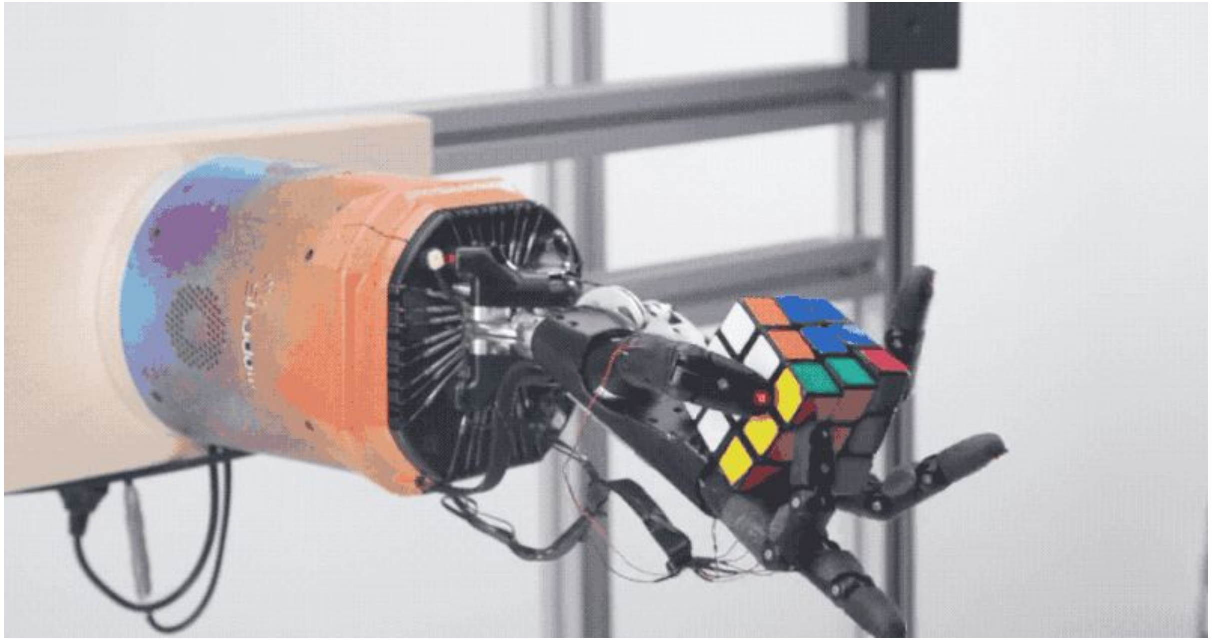
- States
  - How we represent the world
- Actions
  - What our agent can do in the state
- Rewards
  - “Praise” our agent when it does what we want, strengthen its policy
  - “Punish” our agent when it makes mistakes, tell it to change its policy

Video Games

Basketball??


the non-linear control policy learned
with the DDPG algorithm enables
robust control of the skill

Robots



What’s the “goal” of RL training?

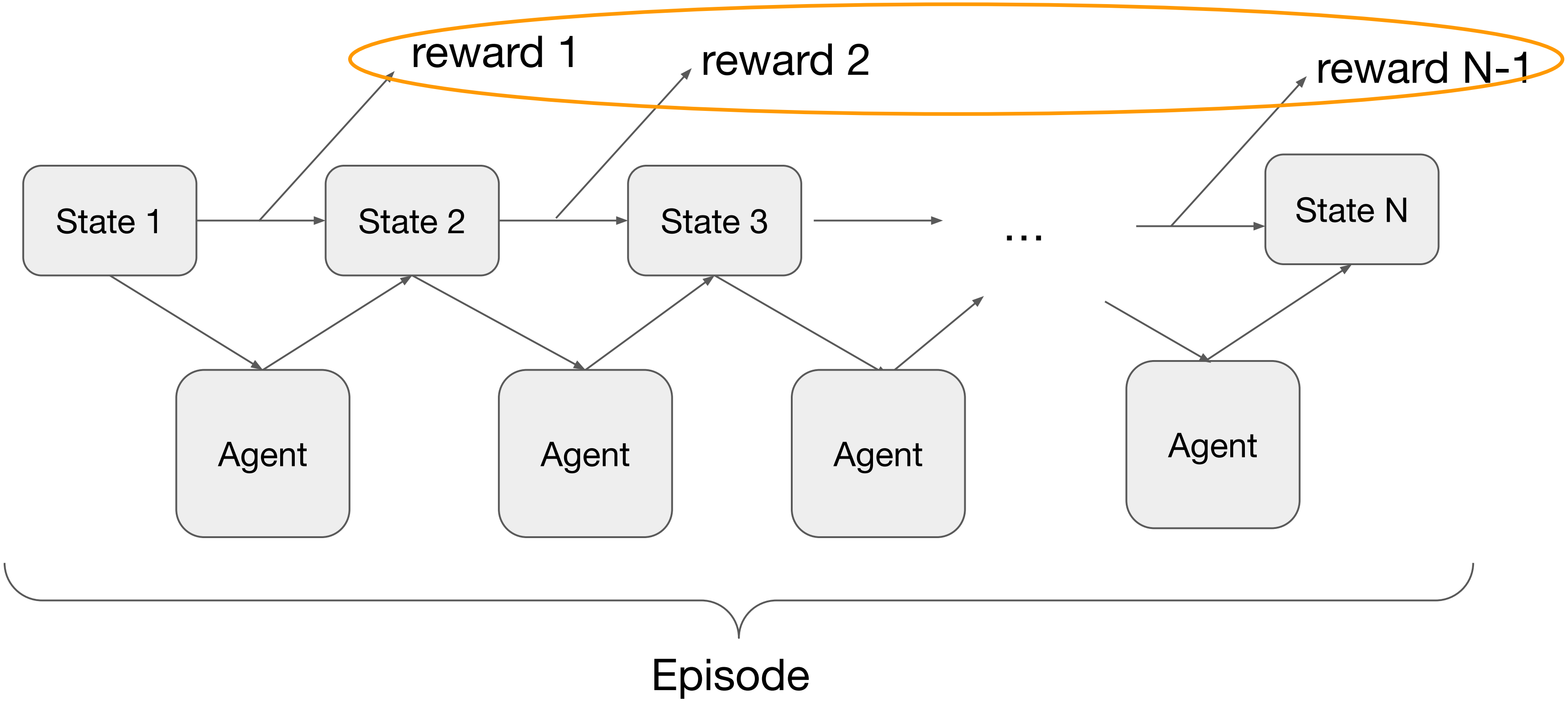
There are algorithmic “goals” in supervised machine learning:

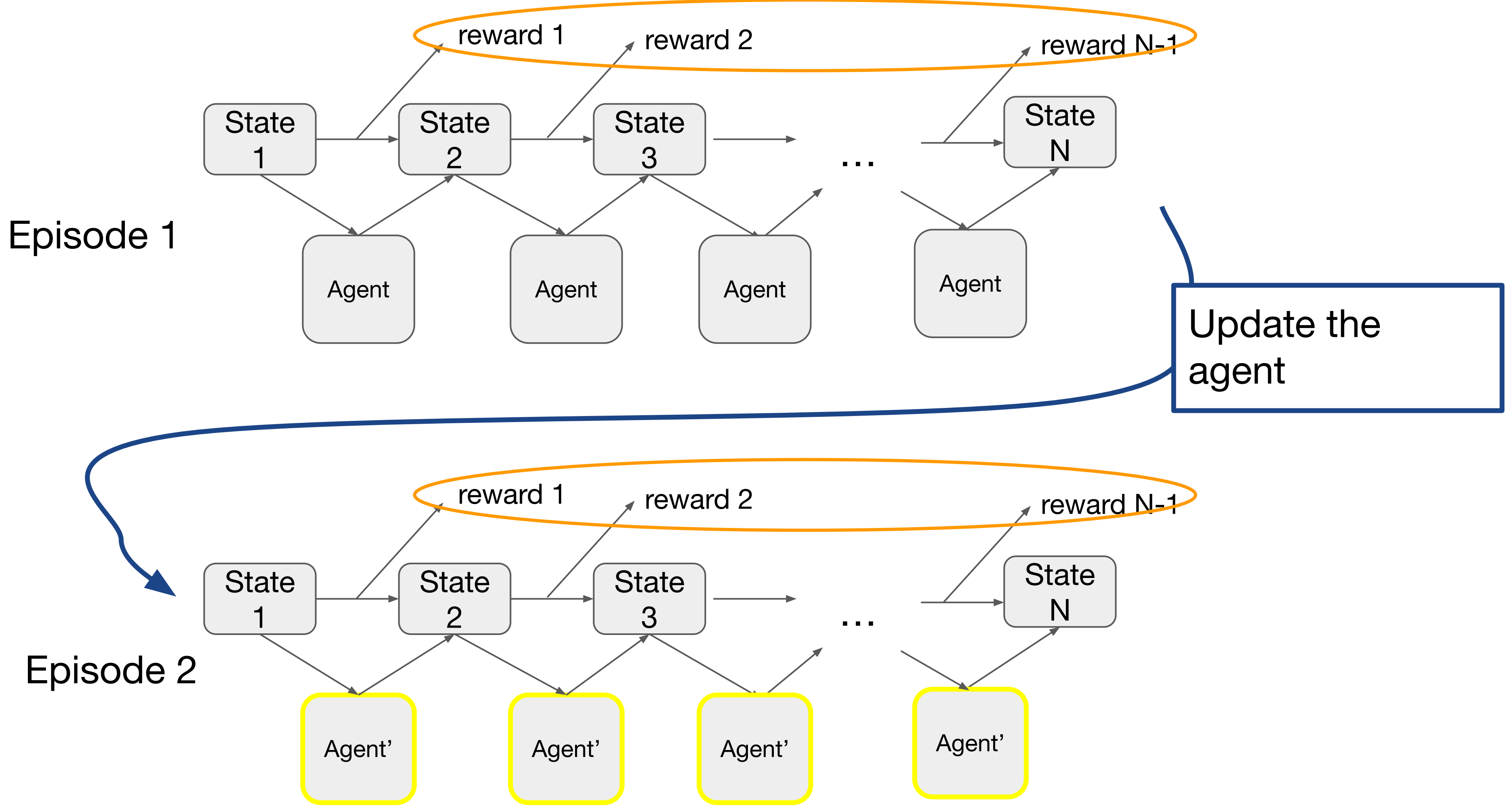
- Minimize our prediction error on observed data (during training)
- Minimize our prediction error on unseen data (generalization)

What’s the goal of RL?

- **Learn a policy** that **maximizes expected cumulative discounted reward**

Cumulative Reward





reward 1
reward 2
reward N-1
State 1
State 2
State 3
...
State N
Episode 1
Agent
Agent
Agent
Agent
Update the agent
reward 1
reward 2
reward N-1
State 1
State 2
State 3
...
State N
Episode 2
Agent'
Agent'
Agent'
Agent'

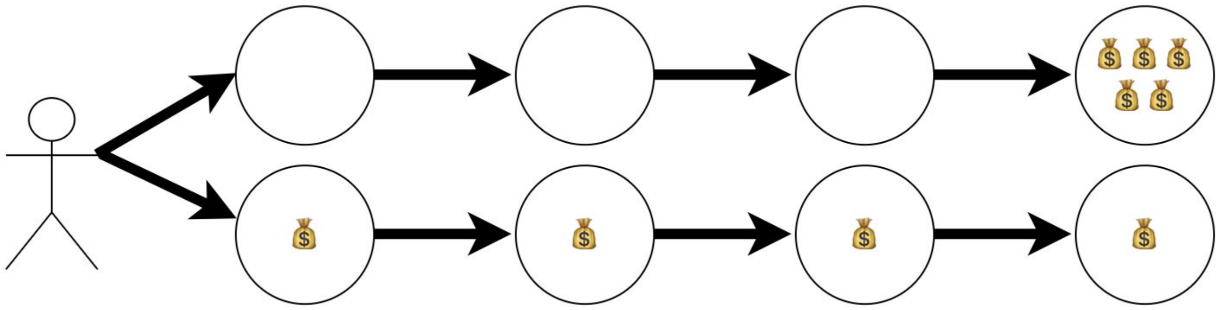
Cumulative Discounted Reward

Cumulative:

Consider future rewards

Discounted:

Immediate rewards might be better than later rewards



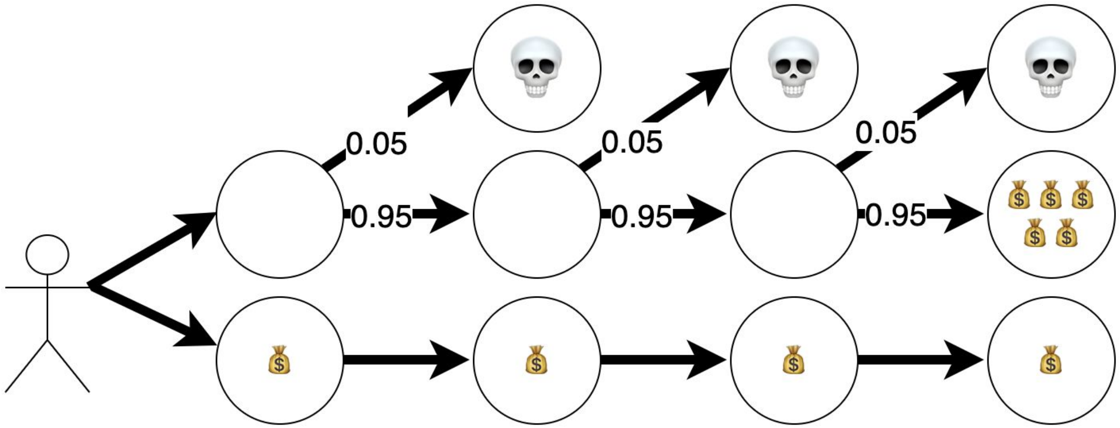
Cumulative Discounted Reward

Cumulative:

Consider future rewards

Discounted:

Immediate rewards might be better than later rewards



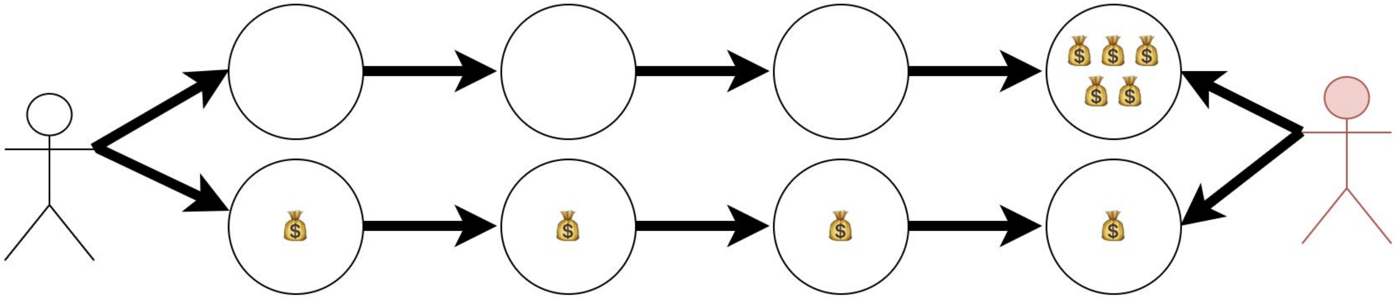
Cumulative Discounted Reward

Cumulative:

Consider future rewards

Discounted:

Immediate rewards might be better than later rewards

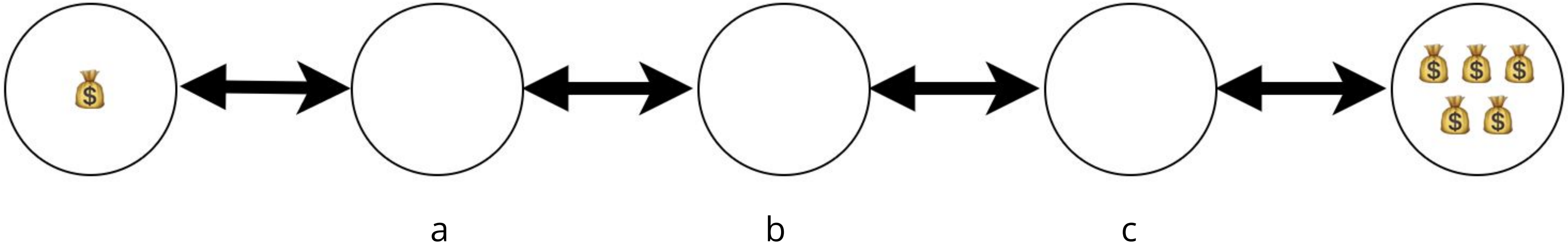


Reward Discounting

Discounted: Immediate rewards might be better than later rewards

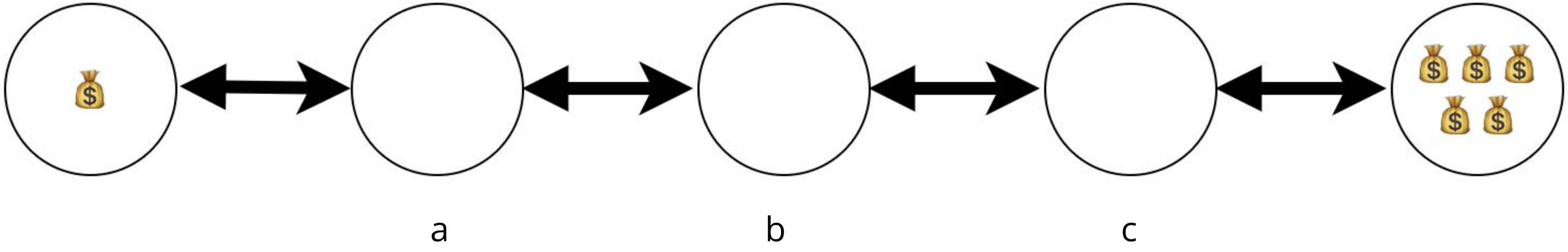
- Each step discounted by a “discount factor” γ
- $0 \geq \gamma \leq 1$, represents how “secure” we are in getting our reward
- In practice usually $\gamma < 1$

Discussion: Which Way Would You Go?



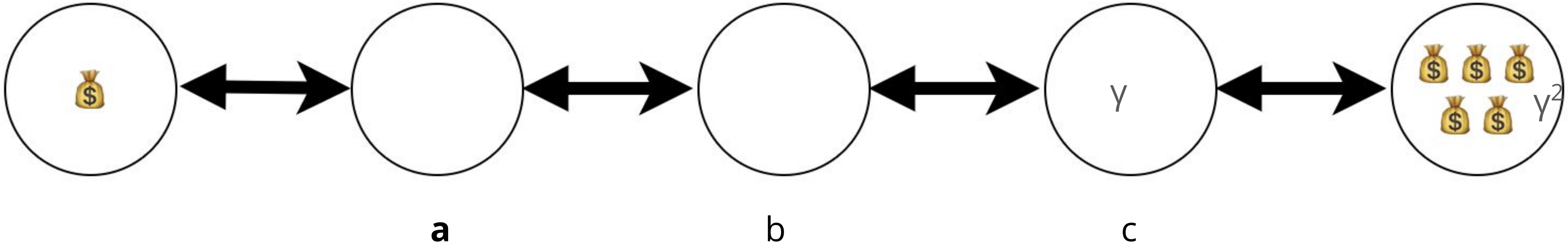
Discussion: Which Way Would You Go?

- Assume determinism
- $\gamma = 1$
- $\gamma = 0.1$
- What γ would make both look equally good to a?



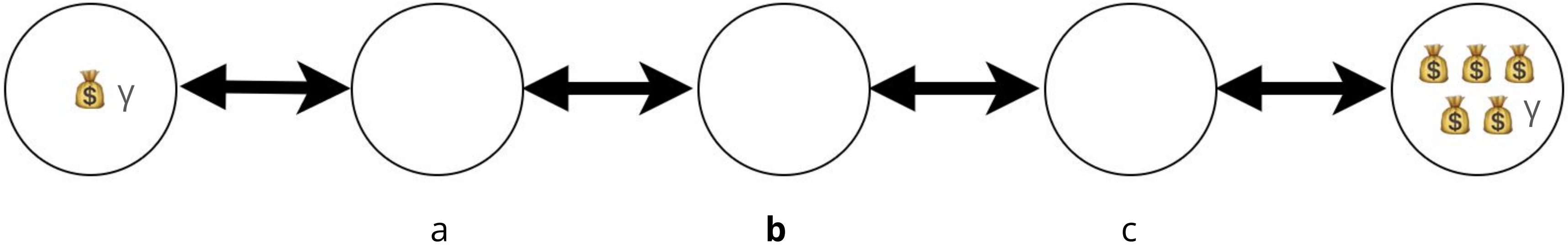
Discussion: Which Way Would You Go?

- Assume determinism
- $\gamma = 1$
- $\gamma = 0.1$
- What γ would make both look equally good to a?



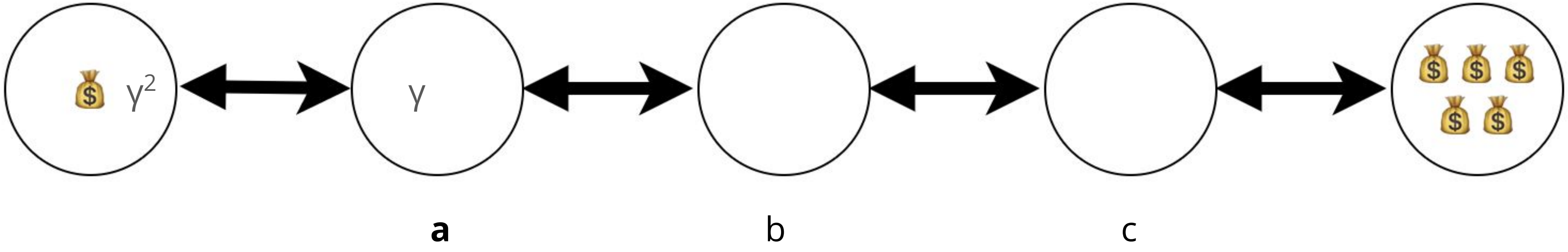
Discussion: Which Way Would You Go?

- Assume determinism
- γ = 1
- γ = 0.1
- What γ would make both look equally good to a?



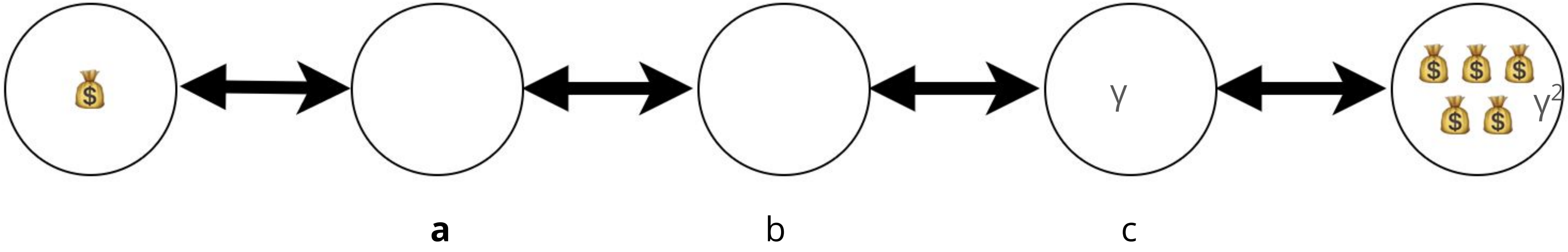
Discussion: Which Way Would You Go?

- Assume determinism
- $\gamma = 1$
- $\gamma = 0.1$
- What γ would make both look equally good to a?



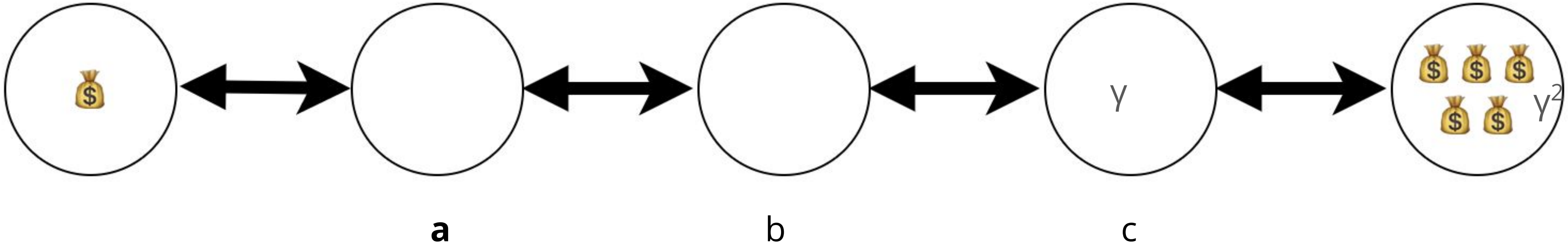
Discussion: Which Way Would You Go?

- Assume determinism
- $\gamma = 1$
- $\gamma = 0.1$
- What γ would make both look equally good to a?



Discussion: Which Way Would You Go?

- Assume determinism
- $\gamma = 1$
- $\gamma = 0.1$
- What γ would make both look equally good to a?
  - $1 = 0 + 5 * \gamma^2$
  - $\gamma = 1/\text{sqrt}(5)$



Infinite Rewards?

How far should we look ahead?

Limit the depth

Issues: Limits applicability, increases state space

Forever, but:

Keep $\gamma < 1$

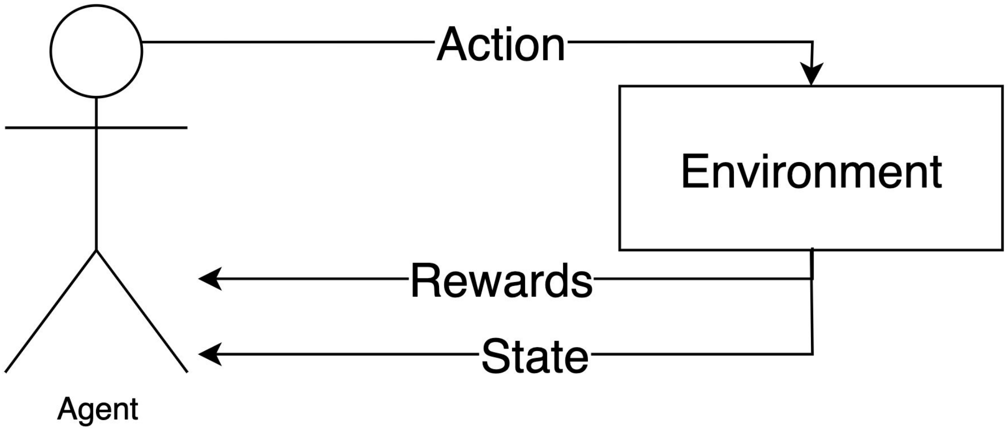
Time penalty

Functions in RL

Utility (aka Value):

Quality

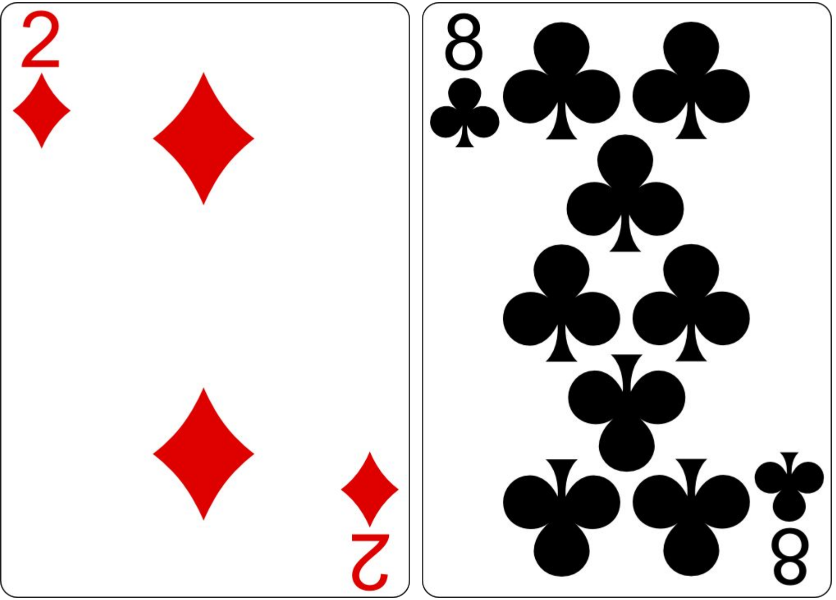
Reward Maximization



$$s_0 \xrightarrow{a_0} r_0 \; s_1 \xrightarrow{a_1} r_1 \; s_2 \xrightarrow{a_2} r_2 \; \ldots$$

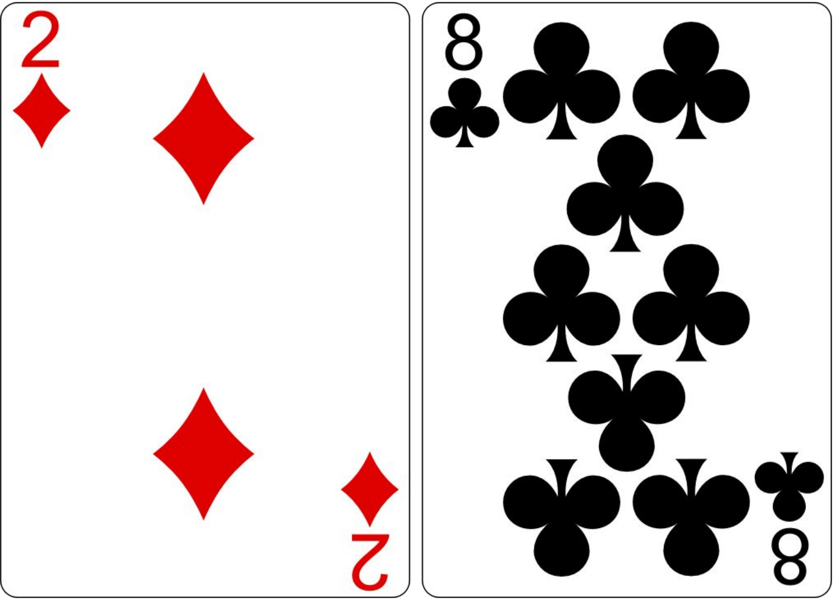
$$\text{maximize } r_0+\gamma r_1+ \gamma^2 r_2+\ldots$$

Blackjack



10

Blackjack



10

-------hit---->

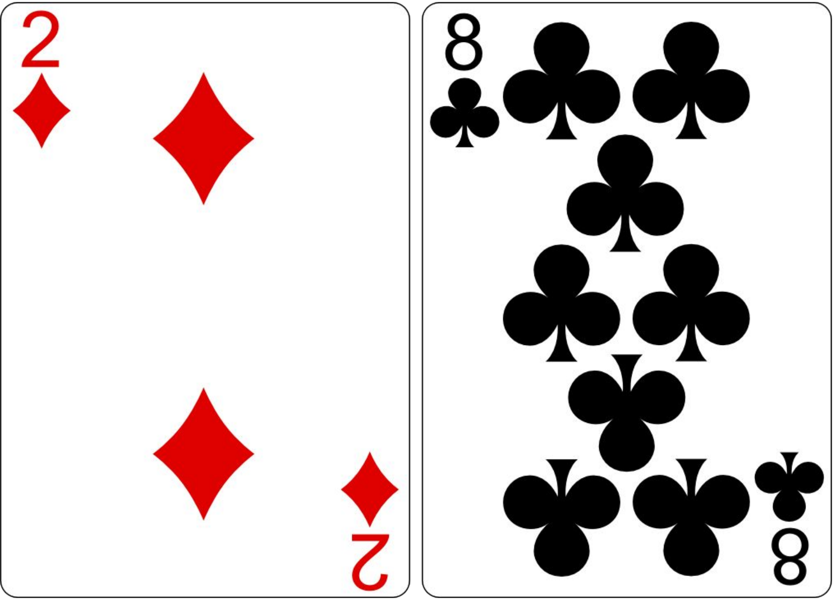
BLACKJACK CHEAT SHEET

techopedia

YOUR HAND	DEALER'S CARD									
	2	3	4	5	6	7	8	9	10	A
8	H	H	H	H	H	H	H	H	H	H
9	H	D/H	D/H	D/H	D/H	H	H	H	H	H
10	D/H	D/H	D/H	D/H	D/H	D/H	D/H	D/H	H	H
11	D/H	D/H	D/H	D/H	D/H	D/H	D/H	D/H	D/H	D/H
12	H	H	S	S	S	H	H	H	H	H
13	S	S	S	S	S	H	H	H	H	H
14	S	S	S	S	S	H	H	H	H	H
15	S	S	S	S	S	H	H	H	R/H	H
16	S	S	S	S	S	H	H	R/H	R/H	R/H
17	S	S	S	S	S	S	S	S	S	S
A,2	H	H	H	D/H	D/H	H	H	H	H	H
A,3	H	H	H	D/H	D/H	H	H	H	H	H
A,4	H	H	D/H	D/H	D/H	H	H	H	H	H
A,5	H	H	D/H	D/H	D/H	H	H	H	H	H
A,6	H	D/H	D/H	D/H	D/H	H	H	H	H	H
A,7	S	D/S	D/S	D/S	D/S	S	S	H	H	H
A,8	S	S	S	S	S	S	S	S	S	S
2,2	P/H	P/H	P	P	P	P	H	H	H	H
3,3	P/H	P/H	P	P	P	P	H	H	H	H
4,4	H	H	H	P/H	P/H	H	H	H	H	H
5,5	D/H	D/H	D/H	D/H	D/H	D/H	D/H	D/H	H	H
6,6	P/H	P	P	P	P	H	H	H	H	H
7,7	P	P	P	P	P	P	H	H	H	H
8,8	P	P	P	P	P	P	P	P	P	P
9,9	P	P	P	P	P	S	P	P	S	S
10,10	S	S	S	S	S	S	S	S	S	S
A,A	P	P	P	P	P	P	P	P	P	P

H	hit
S	stand
P	split
D/H	double down if possible, otherwise hit
D/S	double down if possible, otherwise stand
P/H	split if double down after split is possible, otherwise hit
R/H	surrender if possible, otherwise hit

Blackjack



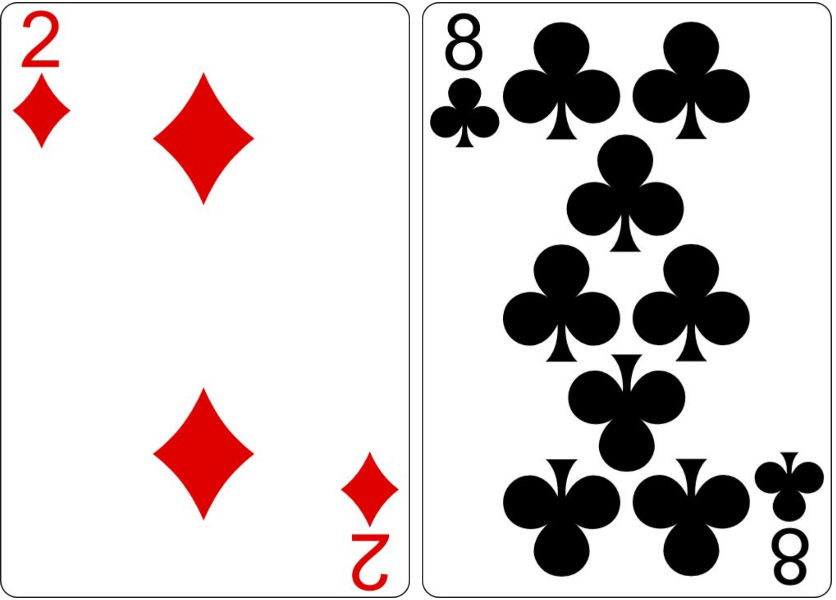
10

-------hit---->

+0

20

Blackjack



10

-------hit---->

20

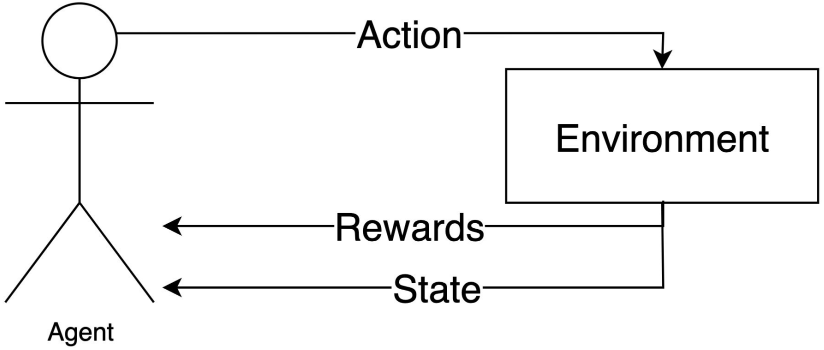
-------stand---->

+1

Blackjack

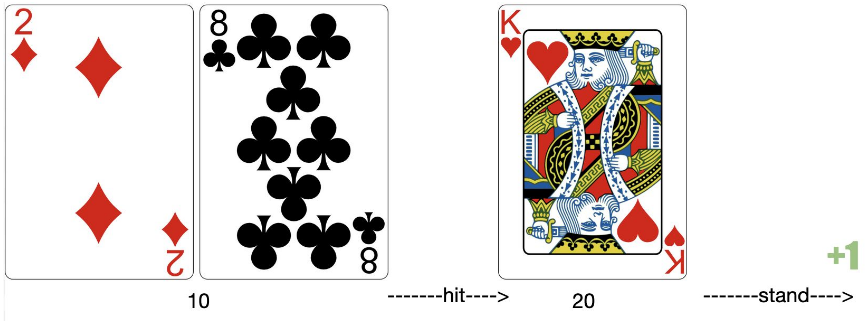
10 –hit→ +0, 20 –stand→ +1, <end>

- What did we learn?



$$s_0 \xrightarrow{a_0} r_0 \; s_1 \xrightarrow{a_1} r_1 \; s_2 \xrightarrow{a_2} r_2 \; \ldots$$

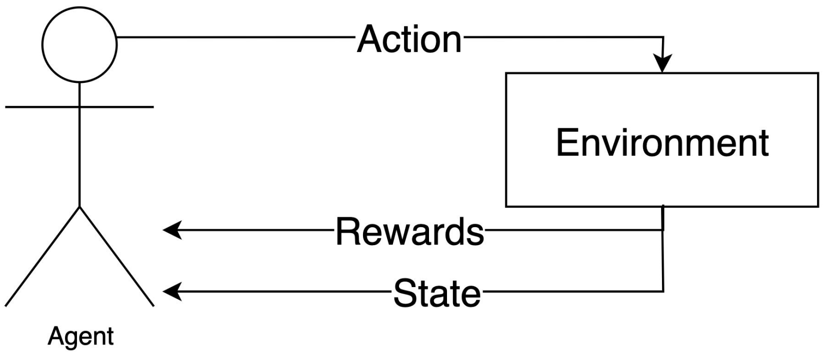
maximize $r_0+\gamma r_1+\gamma^2 r_2+\ldots$



Blackjack

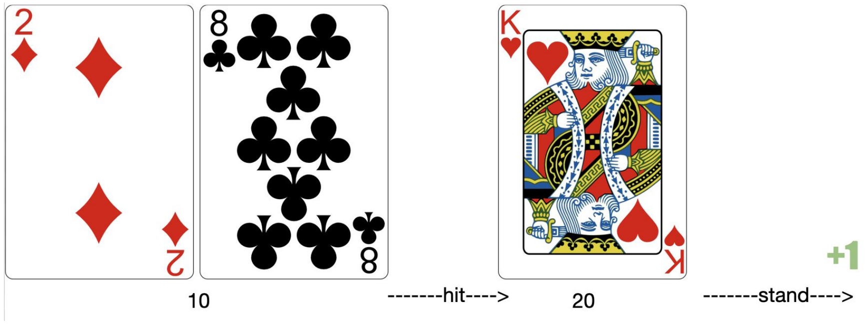
10 –hit→ +0, 20 –stand→ +1, <end>

- What did we learn?
  - Standing on 20 good



$$s_0 \xrightarrow{a_0} r_0 \; s_1 \xrightarrow{a_1} r_1 \; s_2 \xrightarrow{a_2} r_2 \; \ldots$$

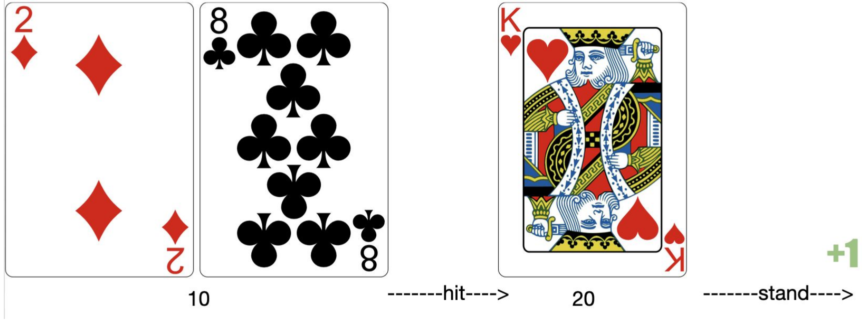
$$\text{maximize } r_0+\gamma r_1+ \gamma^2 r_2+\ldots$$

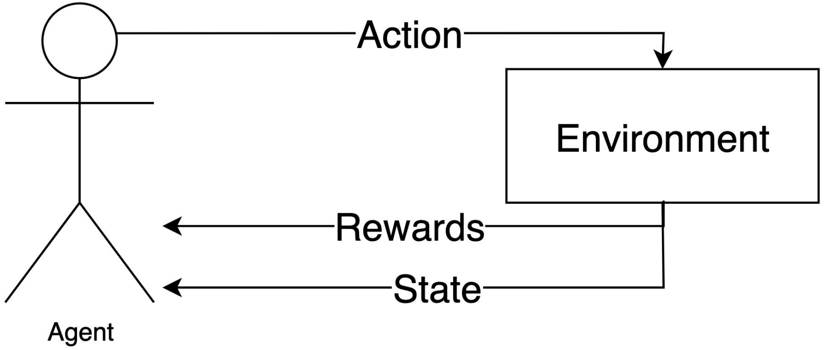


Blackjack

10 –hit→ +0, 20 –stand→ +1, <end>

- What did we learn?
  - Standing on 20 good
    - +1
  - Hitting on 10 good
    - $+1 {}^{*} \gamma = \gamma$





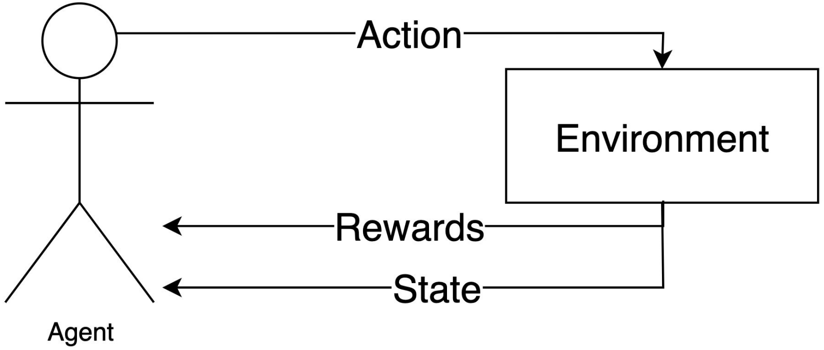
$$s_0 \xrightarrow{a_0} r_0\ s_1 \xrightarrow{a_1} r_1\ s_2 \xrightarrow{a_2} r_2\ \ldots$$

maximize $r_0 + \gamma r_1 + \gamma^2 r_2 + \ldots$

Blackjack

10 –hit→ +0, 20 –stand→ +1, <end>

- What did we learn?
  - Standing on 20 good
    - +1
  - Hitting on 10 good
    - $+1*\gamma=\gamma$
- Issues?



$$s_0 \xrightarrow{a_0} r_0\ s_1 \xrightarrow{a_1} r_1\ s_2 \xrightarrow{a_2} r_2 \ \dots$$

maximize $r_0+\gamma r_1+ \gamma^2 r_2+\dots$

Q Learning

- Learning a Q function that takes in a state and returns the rewards for different actions
  - $Q(s) = [r_{a0}, r_{a1}, r_{a2}, \ldots r_{an}]$
- Update Q function to reflect the cumulative discounted reward
  - $Q(s,a) = r_{s,a} + \gamma * \max_{a'}(Q(s',a'))$
- Take the action that will return the optimal value
  - $A = \mathrm{argmax}_a(Q(s,a))$

State	A_0	A_1	A_2	A_3
S_0	$r_{s0,A0}$	$r_{s0,A1}$	$r_{s0,A2}$	$r_{s0,A3}$
S_1	$r_{s1,A0}$	$r_{s1,A1}$	$r_{s1,A2}$	$r_{s1,A3}$
S_2	$r_{s2,A0}$	$r_{s2,A1}$	$r_{s2,A2}$	$r_{s2,A3}$

The Q-Table

- Can learn Q’, an approximation of the theoretical Q
- Imagine blackjack

State (sum total of cards)	stand	hit
2	0	0
3	0	0
…	0	0
21	0	0

The Q-Table

- Can learn Q', an approximation of the theoretical Q
- Imagine blackjack
- Algorithm:
  - Select action a
  - Transition s –a→s'
  - Update Q-table

State (sum total of cards)	stand	hit
2	0	0
3	0	0
…	0	0
21	0	0

The Q-Table

- Can learn Q’, an approximation of the theoretical Q
- Imagine blackjack
- Algorithm:
  - Select action a
  - Transition s –a→s’
  - Update Q-table

State (sum total of cards)	stand	hit
2	-1	-0.745432
3	-0.9997	0.6532
…	…	…
21	1	-1

The Q-Table

- Can learn Q', an approximation of the theoretical Q
- Imagine blackjack
- Algorithm:
  - Select action a
  - Transition s –a→s'
  - Update Q-table
- Argmax(Q) -> policy

State (sum total of cards)	Action
2	hit
3	hit
…	…
21	stand

The Q-Table

- Table that tells us the “quality” of an action at different states
- Q(s,a) -> cumulative discount reward of applying action a at stat s
- Can learn Q’, an approximation of the theoretical Q
- Imagine blackjack
- Argmax(Q) -> policy

State (sum total of cards)	Action
2	hit
3	hit
…	…
21	stand

The Q-Table

- Table that tells us the “quality” of an action at different states
- Q(s,a) -> cumulative discount reward of applying action a at stat s
- Can learn Q’, an approximation of the theoretical Q
- Imagine blackjack
- Argmax(Q) -> policy
- Abstracts r(s,a) and T(s,a) into a table

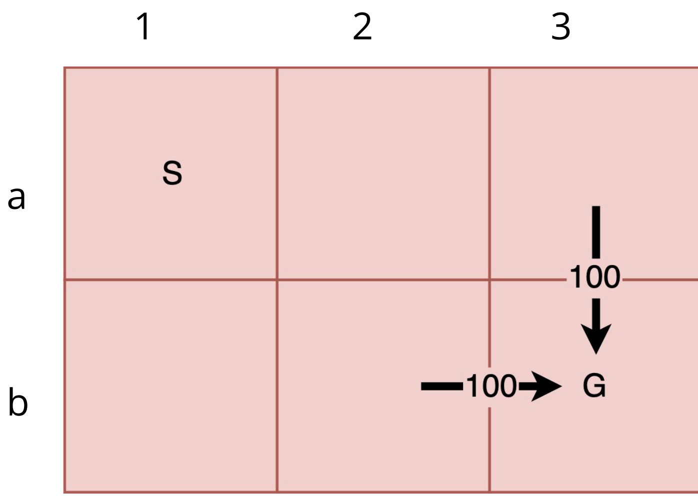
The Q-Table

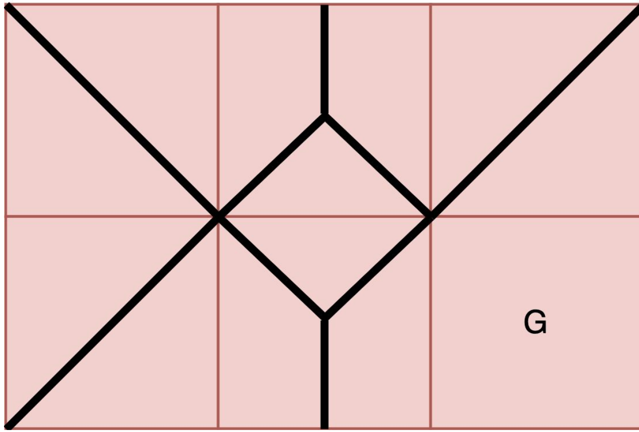
- Table that tells us the “quality” of an action at different states
- Q(s,a) -> cumulative discount reward of applying action a at stat s
- Can learn Q’, an approximation of the theoretical Q
- Imagine blackjack
- Argmax(Q) -> policy
- Abstracts r(s,a) and δ(s,a) into a table
- Let’s see this in action

Building a Q-Table: Gridworld

- $\gamma = 0.9$
- Algorithm:
  - Select action a
  - Transition s –a→s’
  - Update Q-table
- $Q(s_t, a_t) = r_t + \gamma\max(Q(s_{t+1}))$

State	Up	Down	Left	Right
a1	0	0	0	0
a2	0	0	0	0
a3	0	0	0	0
b1	0	0	0	0
b2	0	0	0	0

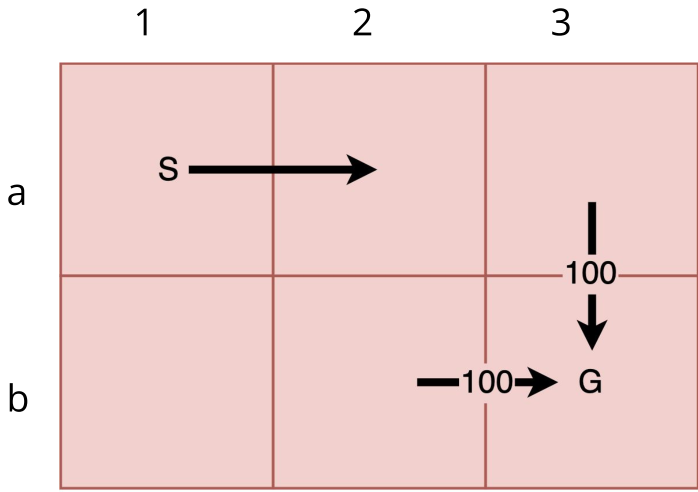


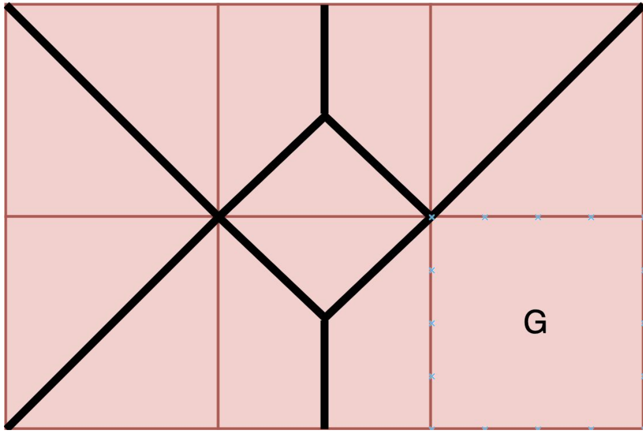


Building a Q-Table: Gridworld

- $\gamma = 0.9$
- Algorithm:
  - Select action a
  - Transition s –a→s’
  - Update Q-table
- $Q(s_t, a_t) = r_t + \gamma \max(Q(s_{t+1}))$

State	Up	Down	Left	Right
a1	0	0	0	0
a2	0	0	0	0
a3	0	0	0	0
b1	0	0	0	0
b2	0	0	0	0

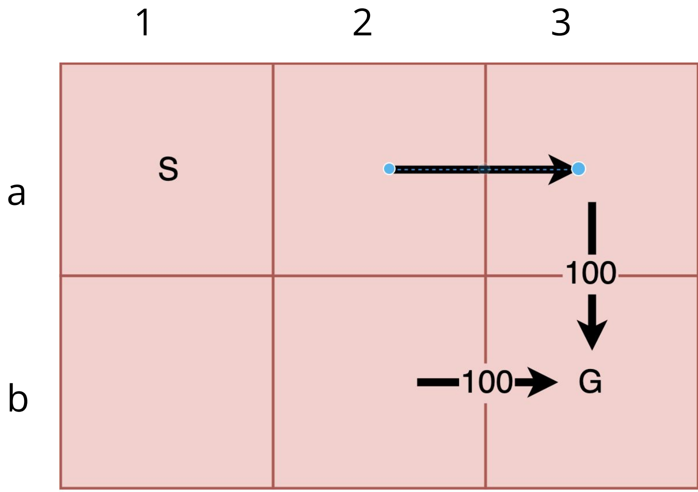


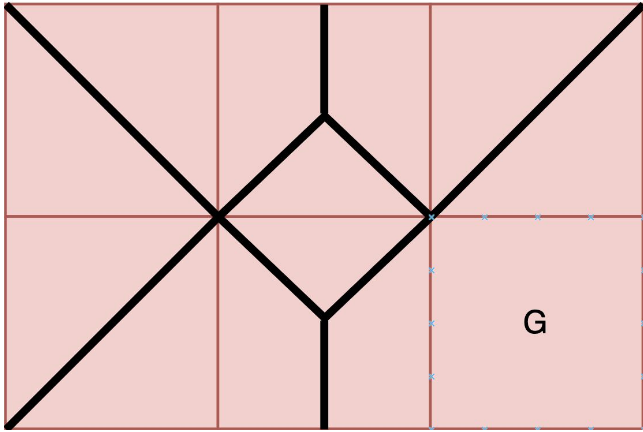


Building a Q-Table: Gridworld

- $\gamma = 0.9$
- Algorithm:
  - Select action a
  - Transition s –a→s’
  - Update Q-table
- $Q(s_t, a_t) = r_t + \gamma \max(Q(s_{t+1}))$

State	Up	Down	Left	Right
a1	0	0	0	0
a2	0	0	0	0
a3	0	0	0	0
b1	0	0	0	0
b2	0	0	0	0

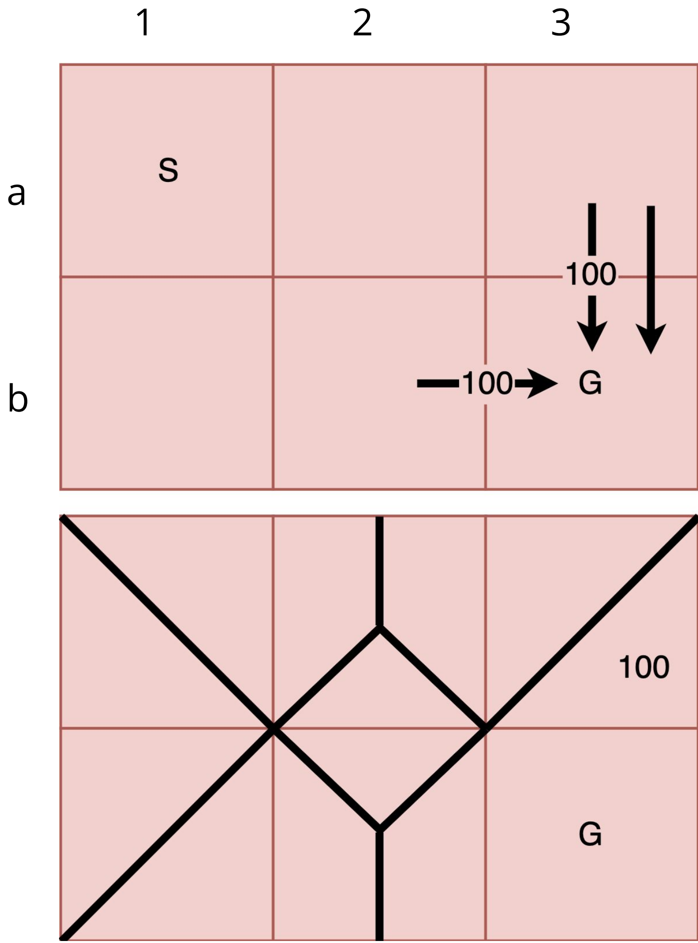




Building a Q-Table: Gridworld

- $\gamma = 0.9$
- Algorithm:
  - Select action a
  - Transition s –a→s'
  - Update Q-table
- $Q(s_t, a_t) = r_t + \gamma \max(Q(s_{t+1}))$

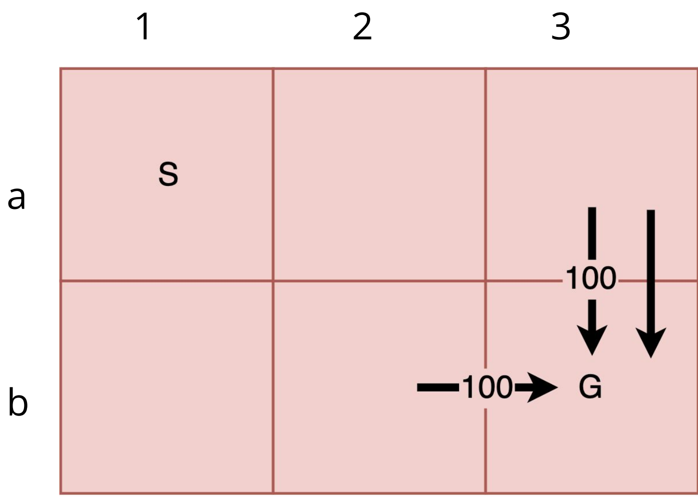
State	Up	Down	Left	Right
a1	0	0	0	0
a2	0	0	0	0
a3	0	100	0	0
b1	0	0	0	0
b2	0	0	0	0

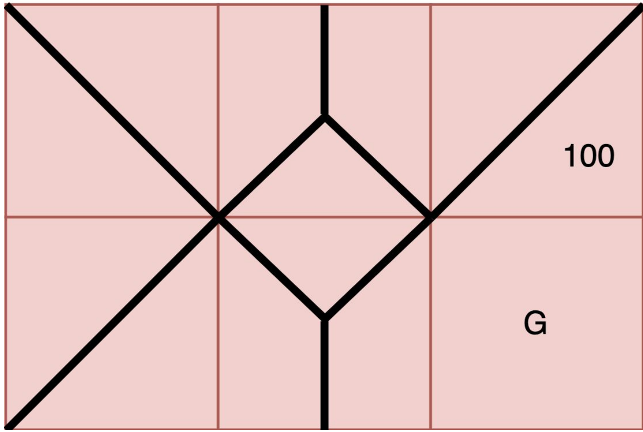


Building a Q-Table: Gridworld

- $\gamma = 0.9$
- Algorithm:
  - Select action a
  - Transition s –a→s’
  - Update Q-table
- $Q(s_t, a_t) = r_t + \gamma \max(Q(s_{t+1}))$

State	Up	Down	Left	Right
a1	0	0	0	0
a2	0	0	0	0
a3	0	100	0	0
b1	0	0	0	0
b2	0	0	0	0

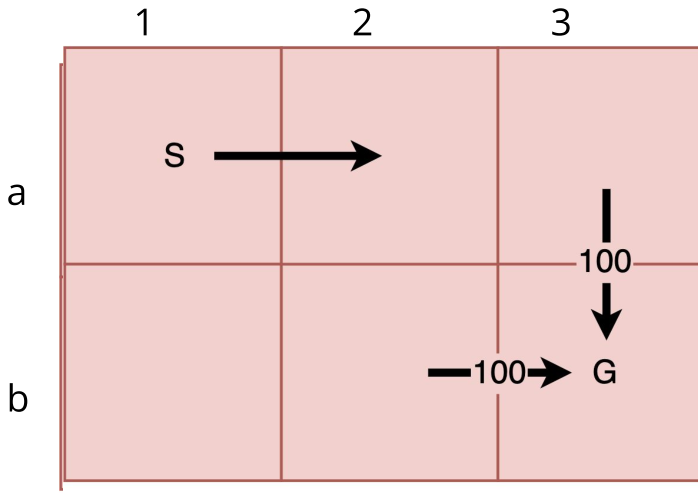


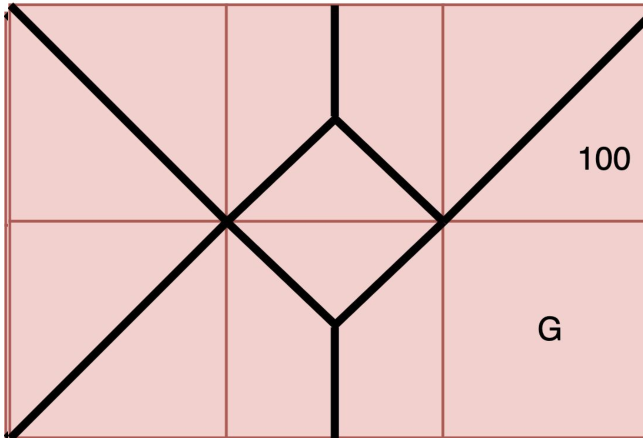


Building a Q-Table: Gridworld

- $\gamma = 0.9$
- Algorithm:
  - Select action a
  - Transition s –a→s'
  - Update Q-table
- $Q(s_t, a_t) = r_t + \gamma \max(Q(s_{t+1}))$

State	Up	Down	Left	Right
a1	0	0	0	0
a2	0	0	0	0
a3	0	100	0	0
b1	0	0	0	0
b2	0	0	0	0

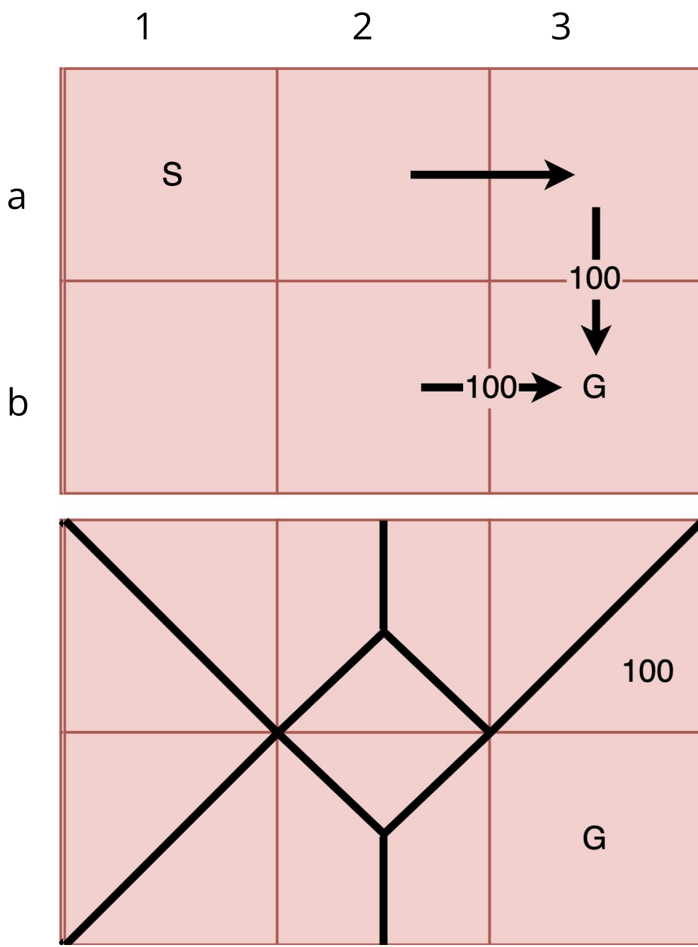




Building a Q-Table: Gridworld

- $\gamma = 0.9$
- Algorithm:
  - Select action a
  - Transition s –a→s'
  - Update Q-table
- $Q(s_t, a_t) = r_t + \gamma \max(Q(s_{t+1}))$

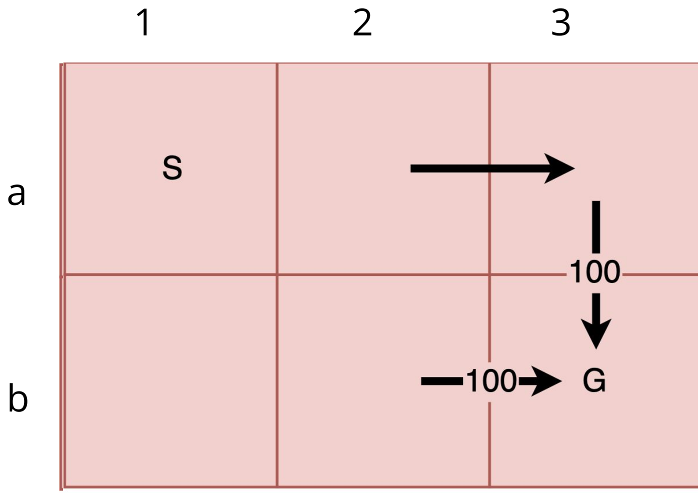
State	Up	Down	Left	Right
a1	0	0	0	0
a2	0	0	0	0
a3	0	100	0	0
b1	0	0	0	0
b2	0	0	0	0

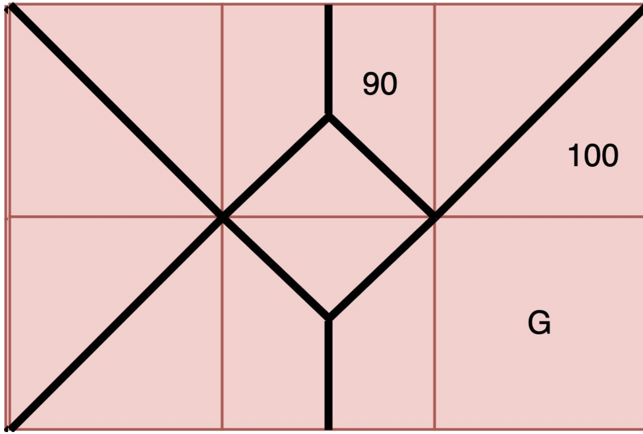


Building a Q-Table: Gridworld

- $\gamma = 0.9$
- Algorithm:
  - Select action a
  - Transition s –a→s’
  - Update Q-table
- $Q(s_t, a_t) = r_t + \gamma \max(Q(s_{t+1}))$

State	Up	Down	Left	Right
a1	0	0	0	0
a2	0	0	0	90
a3	0	100	0	0
b1	0	0	0	0
b2	0	0	0	0

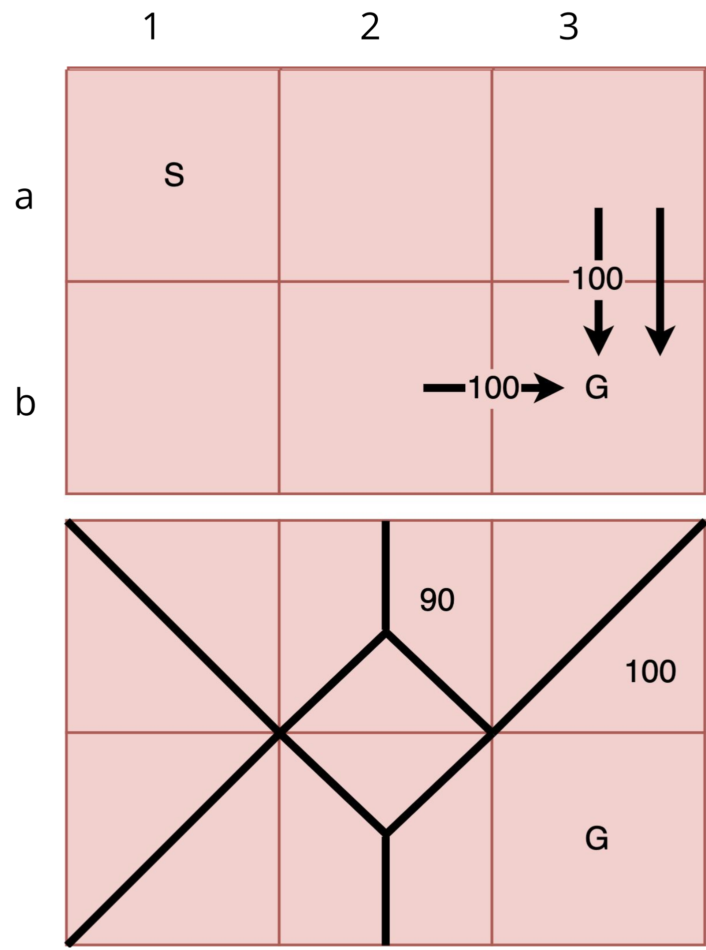




Building a Q-Table: Gridworld

- $\gamma = 0.9$
- Algorithm:
  - Select action a
  - Transition s –a→s’
  - Update Q-table
- $Q(s_t, a_t) = r_t + \gamma \max(Q(s_{t+1}))$

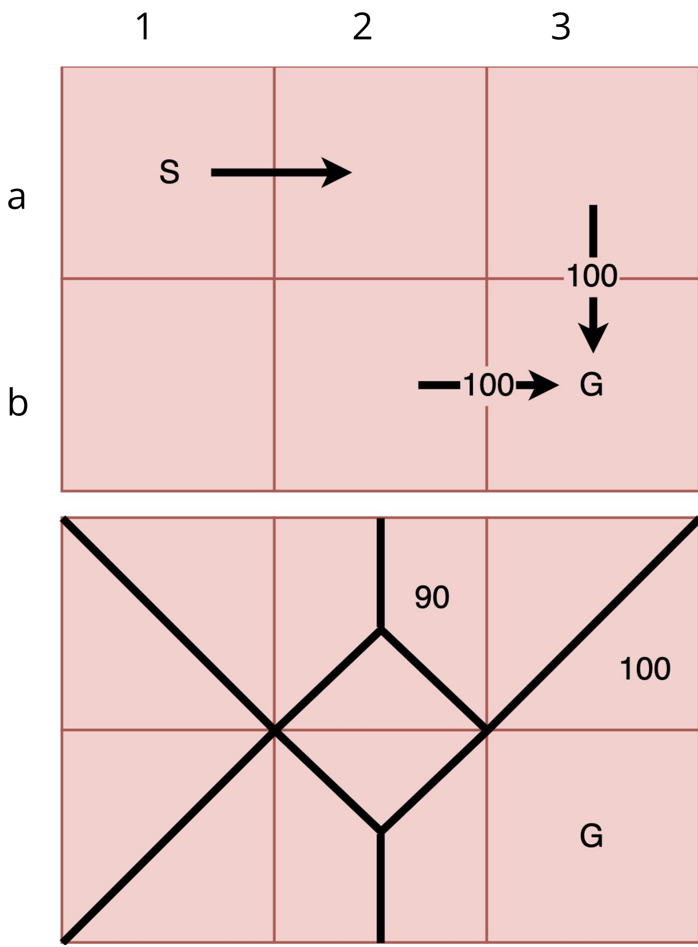
State	Up	Down	Left	Right
a1	0	0	0	0
a2	0	0	0	90
a3	0	100	0	0
b1	0	0	0	0
b2	0	0	0	0



Building a Q-Table: Gridworld

- $\gamma = 0.9$
- Algorithm:
  - Select action a
  - Transition s –a→s'
  - Update Q-table
- $Q(s_t, a_t) = r_t + \gamma \max(Q(s_{t+1}))$

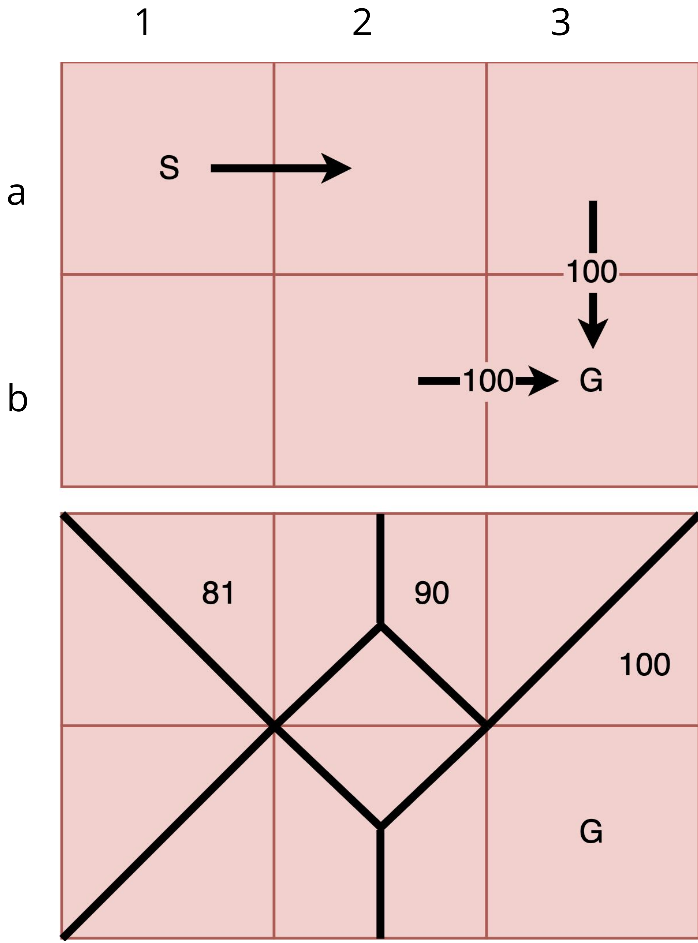
State	Up	Down	Left	Right
a1	0	0	0	0
a2	0	0	0	90
a3	0	100	0	0
b1	0	0	0	0
b2	0	0	0	0



Building a Q-Table: Gridworld

- $\gamma = 0.9$
- Algorithm:
  - Select action a
  - Transition s –a→s’
  - Update Q-table
- $Q(s_t, a_t) = r_t + \gamma \max(Q(s_{t+1}))$

State	Up	Down	Left	Right
a1	0	0	0	81
a2	0	0	0	90
a3	0	100	0	0
b1	0	0	0	0
b2	0	0	0	0



Discussion: Efficiency Per Move

- What is the time complexity of building a minimax tree?

Discussion: Efficiency Per Move

- What is the time complexity of building a minimax tree?
  - $O(b^d)$, or as low as $O(b^{d/2})$ with best case alpha-beta pruning
- What is the time complexity of finding a move here?
  - Time complexity of argmax is O(n)
    - In our case O(b)!
  - Can precompute a policy table and have O(1) lookup at runtime!!

Discussion: Efficiency Per Move

- What is the time complexity of building a minimax tree?
    - $O(b^d)$, or as low as $O(b^{d/2})$ with best case alpha-beta pruning
- What is the time complexity of finding a move here?
    - Time complexity of argmax is $O(n)$
        - In our case $O(b)$!
    - Can precompute a policy table and have $O(1)$ lookup at runtime!!
- The catch: We need to do sufficient training

Markov Decision Process (MDP)

An MDP is a 5-tuple, $\langle S, A, R, P, \rho_0 \rangle$

S	set of valid states	
A	set of valid actions	
$R : S \times A \times S \to \mathbb{R}$	reward function	
$P : S \times A \to \mathcal{P}(S)$	transition probability function	$P(s' \vert s, a)$ is the probability of transitioning into state s' if you start in state s and take action a
$\rho_0 : S \to \mathbb{R}$	initial state distribution	$\rho_0(s)$ is the probability of starting in state s

Terminology

A **state** is a complete description of the state of the world. There is no information about the world which is hidden from the state.

An **observation** is a partial description of a state, which may omit information.

When the agent is able to observe the complete state of the environment, we say that the environment is **fully observed**.

When the agent can only see a partial observation, we say that the environment is **partially observed**.

Terminology

Action space: set of valid actions in a given environment

Discrete action space: only a finite number of moves are available to the agent.

Continuous action space: In continuous spaces, actions are real-valued vectors.

Terminology

A **trajectory** τ is a sequence of states and actions in the world,

$$\tau = (s_0, a_0, s_1, a_1, ...)$$

The very first state of the world, s_0, is randomly sampled from the start-state distribution, sometimes denoted by ρ_0:

$$s_0 \sim \rho_0(\cdot)$$

Infinite-horizon discounted return

$$R(\tau) = \sum_{t=0}^{\infty} \gamma^t r_t.$$

finite-horizon undiscounted return

$$R(\tau) = \sum_{t=0}^{T} r_t.$$